Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aljoša Mrak

Odprtokodna zasnova brezžičnega senzorskega vozlišča

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mira Trebar
Somentor: viš. pred. dr. Robert Rozman

Ljubljana 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Senzorji predstavljajo pomemben vir informacij za krmiljenje energijsko varčnih sistemov v pametni hiši. Obstajajo številne rešitve za povezovanje in krmiljenje naprav z uporabo komercialnih produktov. Nakup takšnih sistemov je pogosto vezan na visoke cene in sistemsko kompatibilnost, ki lahko pomeni določene omejitve pri zagotavljanju uporabnikovih zahtev. Takšnim problemom se je mogoče izogniti z uporabo odprtokodnih rešitev. Kandidat naj v diplomski nalogi predstavi in implementira brezžično senzorsko vozlišče z uporabo mikrokontrolerja ATtiny84 na Arduino platformi in brezžično povezavo na Raspberry Pi strežnik. Vezje naj bo enostavna in čim cenejša prototipna odprtokodna rešitev za pridobivanje senzorskih podatkov temperature in vlage. Podatki in delovanje sistema naj bodo predstavljeni na pametnem telefonu.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aljoša Mrak, z vpisno številko **63110285**, sem avtor diplomskega dela z naslovom:

Odprtokodna zasnova brezžičnega senzorskega vozlišča

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar in somentorstvom viš. pred. dr. Roberta Rozmana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 30. August 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
EPROM	Erasable programmable read only memory	izbrisljiv programirljiv bralni pomnilnik
GCC	GNU Compiler Collection	množica programskih prevajalnikov
IDE	Integrated Development Environment	integrirano razvojno okolje
IR	InfraRed	infrardeča
ISP	In-System Programming	znotraj-sistemsko programiranje
LAN	Local Area Network	lokalno omrežje
MIPS	Million Instructions Per Second	milijon ukazov na sekundo
OSGi	Open Service Gateway initiative	pobuda odprte storitve prehoda
RISC	Reduced Instruction Set Computer	računalnik z zmanjšanim naborom ukazov
SD	Secure Digital	tip spominske kartice
SPI	Serial Peripheral Interface	serijski periferni vmesnik
SRAM	Static Random-Access Memory	statični bralno-pisalni pomnilnik
TV	TeleVision	televizija
UI	User Interface	uporabniški vmesnik
USB	Univeral Serial Bus	univerzalno serijsko vodilo
USI	Universal Serial Interface	univerzalni serijski vmesnik
WiFI	Wireless Fidelity	lokalna brezžična tehnologija

Povzetek

Cilj diplomske naloge je razviti in izdelati majhno, cenovno ugodno odprtokodno vezje. To vezje je sestavljeno iz senzorjev za temperaturo in vlago. Lahko vklaplja in izklaplja luč. V sistem pametne hiše je povezljivo z uporabo brezžičnih tehnologij. Na ta način ga je mogoče upravljati in pregledovati vrednosti senzorjev. Temelji na odprtokodnih tehnologijah in platformah. Centralni sistem je zasnovan na Raspberry Pi in Arduino platformi z odprtokodno programsko opremo. Vsebuje mehanizme za povezovanje z vezjem in s tem upravljanjem tega vezja. Pregled podatkov in upravljanje z vozliščem sta možna preko spletnega vmesnika in pametnega telefona. Razvita programska koda in shema sta objavljeni in prosto dostopni na spletu.

Ključne besede: pametna hiša, odprta koda, openHab, senzorsko vozlišče.

Abstract

The aim of the thesis is to develop and produce a small, affordable open-source circuit. This circuit consists of a temperature and humidity sensor. It can switch a light on and off. It may be linked to the smart home system using wireless technologies. In this way it is possible to manage and inspect values of sensors. It is based on Open Source technologies and platforms. The central system is based on the Raspberry Pi and Arduino platform with open-source software. It contains mechanisms for connecting with the circuit, and thus the management of this circuit. Review of data and management of the node are possible via the web interface and smart phone. The developed software code and scheme are published and freely accessible on the web.

Keywords: smart house, open source, openHab, sensor node.

Poglavje 1

Uvod

Ideja o inteligentnem glasu v naši hiši, ki vedno ve kaj potrebujemo in to naredi namesto nas ob pravem trenutku, izvira že iz 60-ih let [1]. Danes to postaja resničnost. S pametnimi napravami, kot so telefoni, televizije, avtomobili, kuhinjski aparati in žarnice, je le še vprašanje časa, kdaj bo pametna hiša prišla v vsak dom.

Pametna hiša pomeni inteligentno upravljanje naprav v domu s pomočjo senzorjev. Naprave so lahko luči, ključavnice, alarmni sistemi, televizor, kavni avtomati ali celo jacuzziji. Senzorji pa so lahko enostavni termometri, senzorji gibanja ali pa kamere, ki prepoznavajo obraze in razpoznavajo čustva.

S to problematiko so se začela ukvarjati tudi velika podjetja kot so Google, Apple, Microsoft, Samsung itd. Obstajajo komercialne rešitve, ki so zelo drage in zaprte. Vključujejo popoln nadzor nad razsvetljavo, ogrevanjem, varnostjo itd. ter uporabnikom ne omogočajo dodajanja lastnih naprav v sistem.

Zelo pomembna je odprtokodna skupnost, ki si prizadeva izboljšati obstoječe rešitve in jih ponuditi brezplačno. Uporabnikom posredujejo odprtokodne pijače (kjer so navodila za pripravo napitka prosto dostopna in jih lahko kdorkoli poljubno spreminja), avtomobile, 3D tiskalnike, elektroniko (Arduino, OpenRISC, Openmoko, Tinkerforge), programsko opremo (Linux, GNU compiler collection itd.). Vse to lahko uporabnik spreminja, prilagaja in si tudi sam izdela. Zaradi vseh teh lastnosti so taki produkti velikokrat boljši, cenejši, bolj prilagodljivi in imajo večjo podporo skupnosti z raznimi nasveti in predlogi.

Obstajajo odprtokodne rešitve v povezavi s pametno hišo, kot so openHab, Eclipse SmartHome, WOSH Framework itd. To so odprotokodna programska

ogrodja, ki omogočajo avtomatizacijo pametne hiše. So neodvisna od tehnologije in proizvajalca. Njihov namen je zagotavljati enoten dostop do naprav in informacij ter olajšati različne oblike interakcije z njimi [2]. Navadno tečejo na osebnih računalnikih in za komunikacijo z napravami uporabljajo namensko strojno opremo kot so Wi-Fi (Wireless Fidelity) moduli, IR (infrared) sprejemniki, LAN (Local Area Network) kartice, serijska vrata, SPI (Serial Peripheral Interface) vodilo itd. Pred uporabo jih je potrebno še nastaviti, kar pa ne zna vsak. Poleg tega so potrebna še vozlišča. To so manjša vezja nameščena po domu na katera so povezani senzorji in aktuatorji. Kot vozlišča se uporabljajo specifični izdelki posameznih proizvajalcev (Philips Hue, Epson Projector, Samsung TV itd.) ali zaprti sistemi, ki so največkrat dragi in specifični. To bi rešili z odprtokodnim ogrodjem vozlišča ali naborom vozlišč, s katerimi bi opremili hišo. S pomočjo obstoječih odprtokodnih ogrodij (na primer openHab) bi jih povezali v celoto in tako naredili pametno hišo.

Opisani problem je rešen z zasnovo odprtokodnega vozlišča, ki je majhno, cenovno ugodno in enostavno za namestitev v domu. Vsakdo ga lahko uporablja, poljubno spreminja, nadgrajuje in prilagaja.

Izdelali smo testni primer vozlišča. Sestavljeno je iz mikrokrmilnika in brezžičnega vezja. Upravlja luč, meri temperaturo in vlago. V sistem je povezano z odprtokodnim ogrodjem openHab. Na voljo je grafični vmesnik, ki je dostopen preko spletnega brskalnika in mobilne naprave z namensko aplikacijo. Namestili smo ga na Raspberry Pi in dodali strojno in programsko opremo za komunikacijo z vozliščem. Pri programski opremi vozlišča, je bila uporabljena knjižnica nRF24Network za upravljanje brezžičnega čipa in dodana koda za povezavo vozlišča z Raspberry Pi.

Poglavje 2

Teoretične osnove

Tehnologija je povsod okoli nas. Počasi se prebija tudi v naš dom. V vsakem domu že najdemo računalnik, televizor, telefon, kamero. Vedno več srečamo tudi požarnih senzorjev, alarmnih sistemov itd. Če bi vse te naprave povezali med seboj, bi imeli na voljo ogromno količino podatkov. Te podatke lahko izkoristimo za inteligentno krmiljenje naprav v našem domu. Tako lahko marsikatera dela avtomatiziramo, pospešimo in optimiziramo. Kar sovpada z današnjim vedno hitrejšim tempom življenja.

2.1 Pametna hiša

Pametna hiša je sistem, ki zna izkoristiti in upravljati z vsemi napravami, ki jih ima na voljo. To pomeni, da s pomočjo senzorjev zajema podatke iz okolice in na podlagi tega inteligentno krmili naprave v domu, kot so luči, ogrevanje, senčila, ključavnice in drugo. Na primer ko oseba vstopi v prostor, se lahko luč vklopi, ali pa se pralni stoj zažene ob cenejši električni energiji.

Ko pripeljemo računalnik v dom, lahko marsikaj naredimo bolj optimalno kot sicer. To lahko zmanjša obratovalne stroške hiše. Luči svetijo le, ko je nekdo v prostoru, senčila, ogrevanje in klimatske naprave delujejo složno, tako da ogrejemo dom najprej s soncem in šele nato se vklopi centralna kurjava. Takšno delovanje definirajo sklopi pravil, ki na podlagi meritev iz senzorjev, upravljajo naprave kot so posamezne luči, posamezna senčila. Primer take obstoječe komercialne rešitve je Loxone [11].

2.2 Strojna oprema pametne hiše

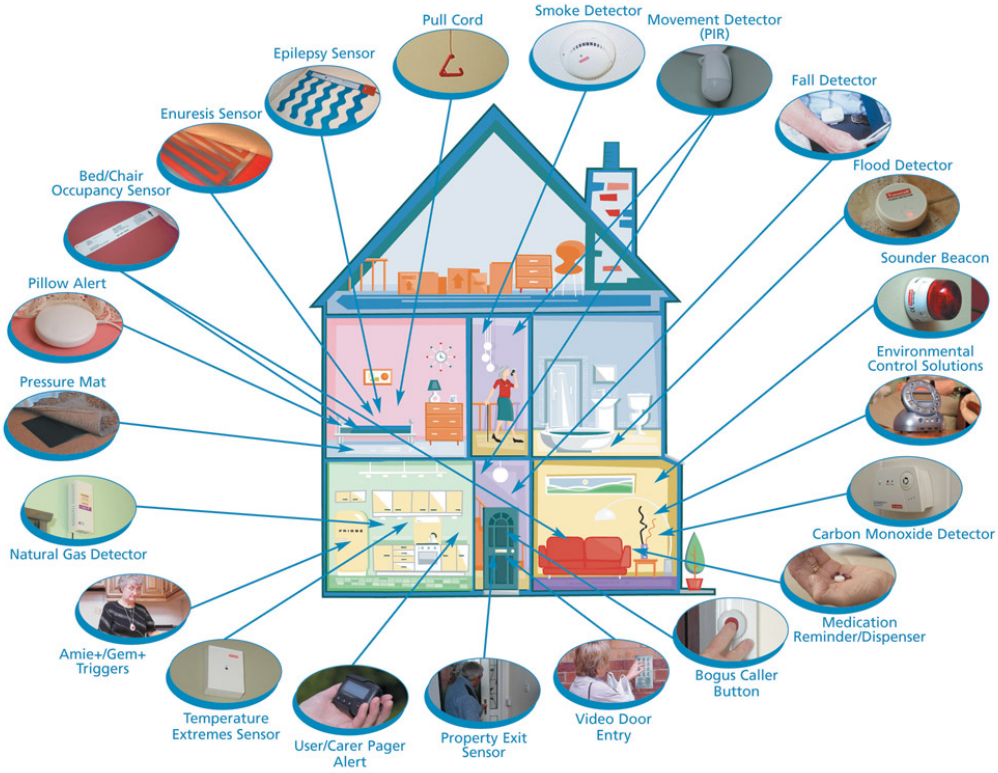
Strojna oprema pametne hiše se deli na več delov. Glavni del je centralna enota. Ponavadi je to manjša naprava, ki ima nekaj posebnih komponent, ki izkoriščajo različne tehnologije in omogočajo povezovanje z ostalimi napravami (na primer LAN adapter, WiFi adapter itd.). To je lahko navaden računalnik ali specializirana enota za določeno tehnologijo in protokol (na primer Z-wave ali Loxone) ali interaktivna naprava, kjer avtomatizacija hiše ni primarna funkcija. Na sliki 2.1 so prikazani ti trije primeri.

Slika 2.1: Primeri centralnih enot. Na levi je Raspberry Pi [17], v sredini je centralna enota za Z-Wave [18], in na desni je Jibo [19].

Poleg tega potrebujemo naprave, ki jih lahko upravljamo (Slika 2.2). To so lahko luči, senčila, ogrevalni sistem, kavomati in druge. Vsaka mora imeti že vgrajen način komunikacije z ostalim sistemom (na primer Philips Hue) ali pa je to potrebno dodati (na primer vklapljanje luči z relejem).

Za avtonomno upravljanje s temi napravami pa so potrebni še senzorji (Slika 2.3). Na ta način se lahko sistem odziva na spremembe v okolju. Ko oseba vstopi v prostor in je temno, se prižge luč. Glede na primarni namen pametne hiše, potrebujemo različne senzorje. V vsakodnevnem življenju si želimo olajšati in poenostaviti vsakdanja opravila. V takem primeru potrebujemo enostavnejše senzorje, kot so senzorji za temperaturo, vlago, dež, veter, razne kamere, CO_2 detektorji, senzorji za vlom itd. Za pomoč starejšim osebam in osebam s fizičnimi nezmožnostmi pa želimo zagotavljati mobilnost in spremljanje zdravja. V takem primeru želimo imeti senzorje padca, epilepsije, detektorje gibanja, analize krvi, glasovno aktiviran klic na pomoč itd.

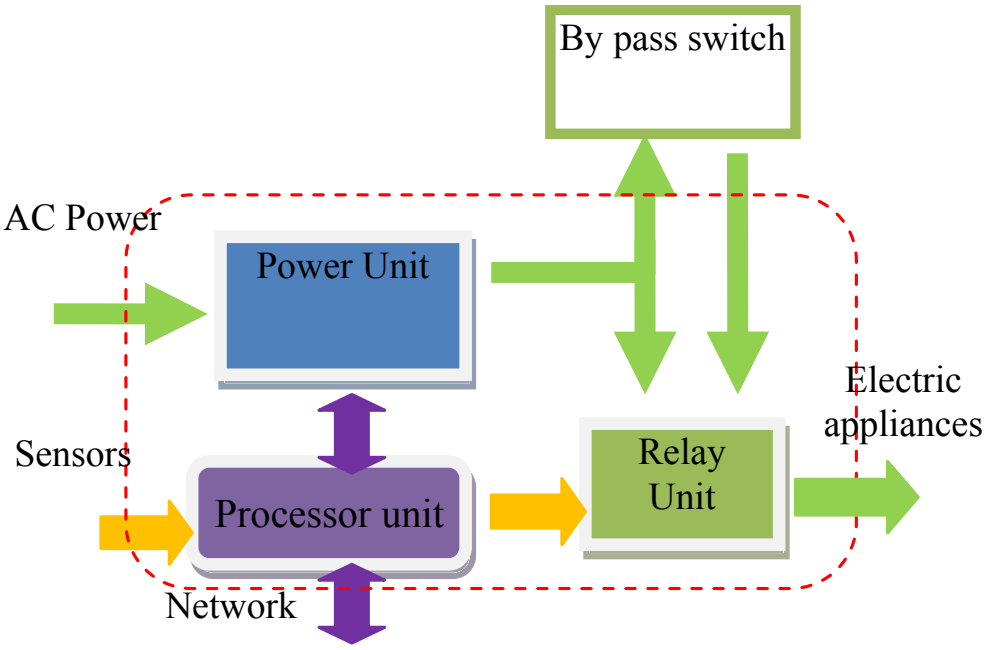
Slika 2.2: Naprave v pametni hiši: Žarnica Philips Hue [9] (levo), Goji pametna ključavnica [10] (desno).



Slika 2.3: Nabor senzorjev v pametni hiši [20]

2.3 Obstoječa rešitev

Hsien-Tang Lin je v svojem delu "Implementing Smart Homes with Open Source Solutions"[3] predlagal arhitekturo vozlišča na sliki 2.4. Sestavlja jo procesna enota, napajalni sistem, premostitveno stikalo in rele enota. Procesna enota upravlja vozlišče in povezuje vse komponente med seboj. Premostitveno stikalo je mehanski del, ki deluje kot rezervni mehanizem v primeru, da vozlišče ne deluje pravilno (izpad elektrike, prazna baterija, motnje v komunikaciji itd.). Procesna enota s pomočjo rele enote vklaplja in izklaplja naprave, ki so povezane na njo.



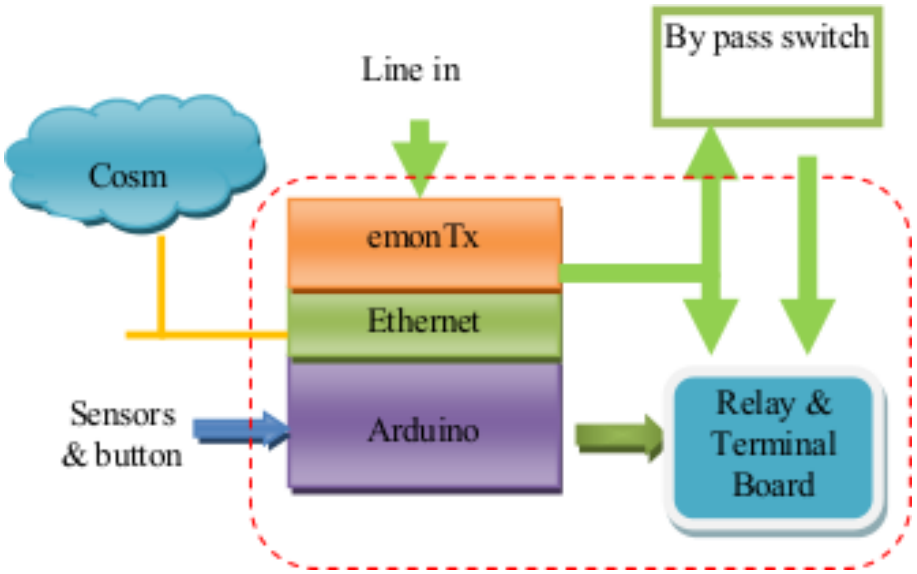
Slika 2.4: Predlagana arhitektura v članku [3].

V članku je omenjeno tudi merjenje porabe električne energije. Včasih je potrebno vedeti kolikšna je poraba razsvetljave v kuhinji ali kolikšna je poraba klimatske naprave med poletjem. Glede na te podatke je mogoče zmanjšati porabo električne energije in tako privarčevati. Te meritve največkrat niso natančne in vse električne naprave v stanovanju niso povezane v pametno hišo. Izkaže se, da meritev ni potrebno opraviti točno, ker je pri porabi zanimivo predvsem razmerje (na primer v kuhinji je dvakrat večja poraba kot v kopalnici). Iz teh meritev

je mogoče razbrati ali naprava deluje in ali deluje pravilno. Tako je vidno ali je žarnica pregorela, ker ni porabe ali druge okvare v primeru prevelike porabe.

Predstavljena je tudi konkretna implementacija na sliki 2.5. Sestavljena je iz prototipnega orodja Arduino, Ethernet nastavka, emonTx nastavka in Cosm oblačne storitve. Ker je vse skupaj le prototip arhitekture, je potrebno narediti nekaj sprememb za dejansko uporabo v sistemu pametne hiše. Komponente, ki so uporabljene, so prevelike in predrage.

Arduino je prototipna platforma in ni namenjena uporabi v produktih. Boljša rešitev je samo mikrokrmilnik brez konektorjev in priključkov. Za povezovanje z ostalimi komponentami sistema pa je lažje uporabiti brezžične tehnologije. Tako ni potrebno predelati celotne infrastrukture po hiši, da je do vsakega vozlišča napeljan ethernet kabel.



Slika 2.5: Konkretna implementacija predlagana v članku [3].

Poglavje 3

Vozlišče

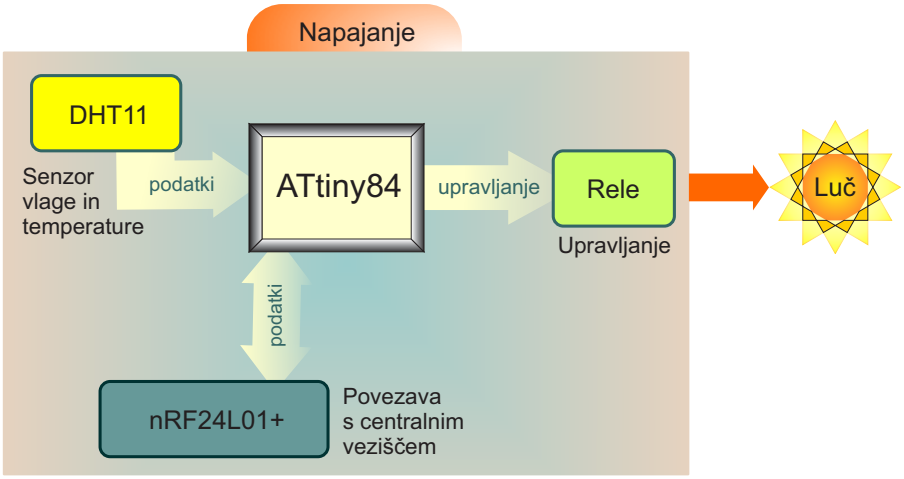
Vozlišče je del pametne hiše, ki zagotavlja interakcijo z okolico. Lahko na njo vpliva (na primer odpira vrata, prižiga/ugaša luči itd.) ali pa jo zaznava (na primer zaznavanje gibanja, merjenje temperature itd.).

V primeru pametne naprave, ki jih je danes vedno več, je vozlišče že del naprave (na primer pametni pomivalni stroj, pametna pečica). V takem primeru moramo poskrbeti samo za pravilno komunikacijo med napravo in ostalim delom pametne hiše. Če pa naprava ni pametna, moramo izdelati celotno vozlišče. Tako vozlišče mora znati upravljati s to napravo in komunicirati s pametno hišo. Komunikacija lahko poteka preko različnih medijev in protokolov.

Najpogosteje se napravo upravlja tako, da ji vključimo ali izključimo napajanje (na primer zjutraj se vklopi napajanje grelnika vode, da nam pripravi topel napitek). Na ta način ne posegamo v samo napravo. Lahko pa spremenimo sámo vezje naprave in speljemo povezave skozi vozlišče. Tako lahko vozlišče poljubno upravlja z napravo (na primer nastavi temperaturo na pečici in čas peke). Tak poseg je drag, zahteven in ni nujno da bo deloval. Za vsak model naprave je potrebno pripraviti svoje vozlišče z drugačno programsko opremo, kar je pa drago in zamudno.

3.1 Zasnova vozlišča

Odprtokodno vozlišče smo zasnovali na osnovi predstavljene rešitve (poglavje 2.3). Potrebno pa je opraviti nekaj sprememb in se približati enostavnejšemu



Slika 3.1: Shematski prikaz vozlišča.

cenovno ugodnemu komercialnemu produktu.

Povezovanje vozlišča preko Etherneta zamenjamo z brezžično tehnologijo. Tako ni potrebno napeljevati dodatnih kablov po domu. Potrebujemo centralni sprejemnik, ki sprejema podatke iz vseh vozlišč. Namesto Cosm oblačne storitve uporabimo openHab in to vključimo v centralni sprejemnik. Arduino moramo zamenjati, saj je le prototipna platforma. Na voljo imamo kar nekaj mikrokrmilnikov. Ravno tako lahko zamenjamo emonTx s samim čipom in tako naredimo vse skupaj ceneje.

Vozlišče je shematsko prikazano na sliki 3.1. Za mikrokrmilnik je uporabljen zelo razširjen AVR, konkretno ATtiny84. V sistem se povezuje brezžično z nRF24l01+. Za merjenje porabe je uporabljen hall senzor. Vozlišče se napaja kar iz omrežja preko transformatorja, usmernika in regulatorja napetosti. Poleg tega pa ima vsako vozlišče svoje senzorje in/ali posamezne priklopljene naprave (releji, stikala itd.).

3.2 Strojna oprema

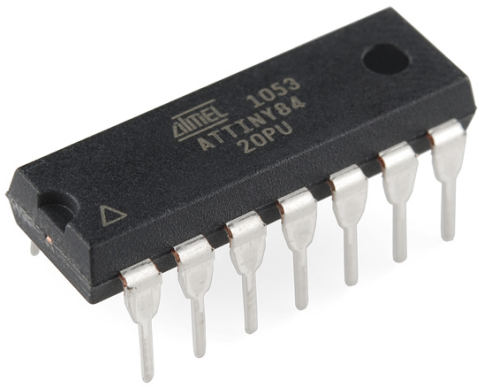
Vozlišče mora biti majhno, enostavno, prilagodljivo, modularno in cenovno ugodno. Sestavljeno je iz mikrokrmilnika, povezovalnega vmesnika (brezžičnega ali žičnega), senzorjev in/ali aktuatorjev in sistema napajanja.

3.2.1 ATtiny84

Atmel proizvaja družino mikrokontrolerjev imenovano AVR [12]. Ta temelji na 8-bitni RISC arhitekturi in je prvi mikrokontroler, ki uporablja bliskovit pomnilnik (ang. flash) na čipu. Razvojna orodja so brezplačno na volja na spletu. Obstaja pa tudi velika skupnost "AVR freaks", kjer je moč dobiti navodila, projekte, orodja itd. Uporabljeni so tudi v prototipni platformi Arduino. Zaradi tega so AVR čipi enostavni za uporabo in primerni v tem in marsikaterem drugem projektu.

Čipi z oznako ATtiny so najmanjši čipi v družini AVR. Ponujajo zmogljivost, energijsko učinkovitost in enostavno uporabo v majhnem pakiranju.

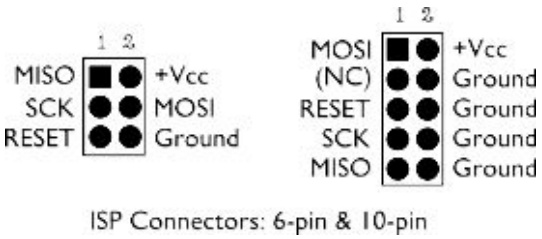
Za to nalogo je najbolj primeren ATtiny84 (slika 3.2) predvsem zaradi fizične velikosti, števila pinov in količine spomina. Je eden izmed bolj razširjenih in podprtih AVR mikrokrmilnikov. Čip deluje na 20 MHz, 1 MIPS na MHz. Ima 14 pinov od teh je 12 vhodno/izhodnih. Na voljo ima 8KB ISP flash spomina, 512B EEPROM, 512B integriran SRAM, univerzalni serijski vmesnik (USI) za komunikacijo z ostalimi komponentami in vmesnik debugWIRE [8]. DebugWIRE je serijski protokol, ki je namenjen razhroščevanju na čipu.



Slika 3.2: ATtiny84 [21].

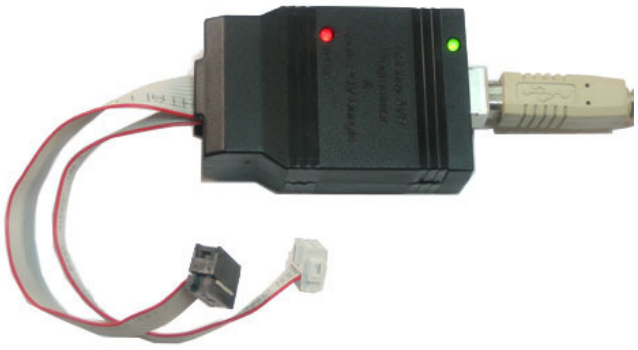
Programiranje

AVR čipi se programirajo preko ISP vmesnika (na sliki 3.3), ki je podoben vodilu SPI (Serial Peripheral Interface). ISP uporablja skupno uro (SCK), serijski izhod (MOSI), serijski vhod (MISO), napajanje in negiran reset, ki je podoben CS (Chip Select). Ko je reset povezan na pozitivno napetost, na čipu teče naš program, ko pa je ozemljen, gre čip v način programiranja in takrat posluša na teh pinih.

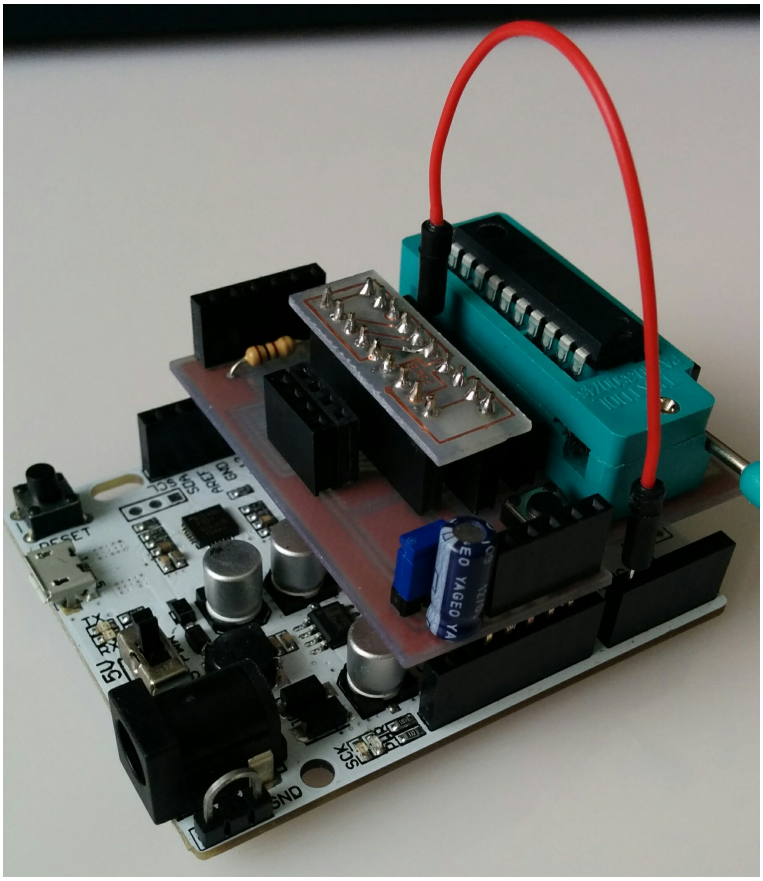


Slika 3.3: 6 in 10-pinski ISP konektor [4].

Če želimo programirati AVR čip potrebujemo namensko programsko opremo in namensko strojno opremo, ki deluje kot posrednik med računalnikom in čipom. Programska oprema AVR IDE je prosto dostopna na spletu. Potrebujemo še ISP programator na primer USBtinyISP na sliki 3.4.



Slika 3.4: USBtinyISP programator [6].

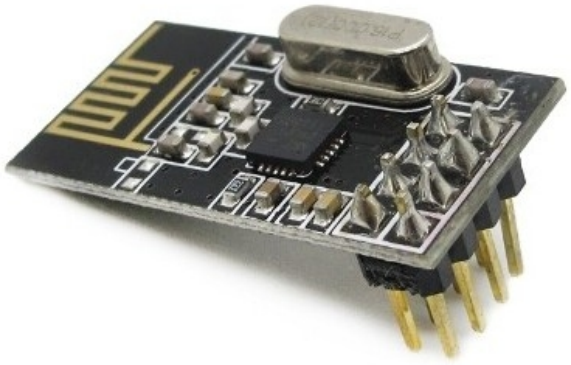


Slika 3.5: Arduino z nastavkom za programiranje AVR čipov.

Namesto tega lahko uporabimo prototipno ploščico Arduino na katero naložimo program ArduinoISP. Tako se Arduino obnaša kot ISP programator. Kot programsko opremo uporabimo Arduino IDE. Ta program prevede in ga pošlje na Arduino. Ta pa ga naloži na čip. Za lažjo povezavo se uporablja Arduino nastavek na sliki 3.5.

3.2.2 nRF24L01+

nRF24L01+ (slika 3.6) je brezžično oddajno/sprejemno integrirano vezje z nizko porabo. Deluje na 2.4 GHz. Komunikacija s čipom poteka preko SPI vodila. Čip strojno podpira 2. plast v ISO/OSI referenčnem modelu oziroma povezovalno plast [7]. Ta skrbi za določanje enote sporočil, obravnavanje napak med dvema



Slika 3.6: nRF24L01+ čip [22].

sosednjima vozliščema, mehanizme dostopa do prenosnega medija in kontrolo pretoka.

Ker je cenovno zelo ugoden, majhen in enostaven za uporabo, je kot nalašč za ta projekt.

3.2.3 Raspberry Pi

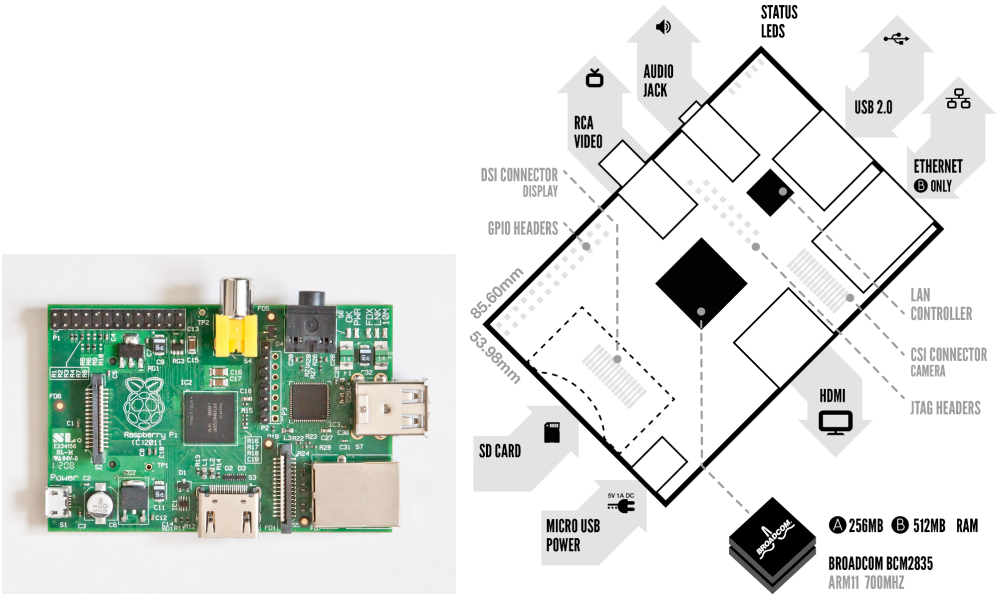
Rasperry Pi je računalnik na eni ploščici v velikosti kreditne kartice (slika 3.7). Razvit je bil z namenom promoviranja računalništva med otroci. Zaradi nizke cene in velikosti se je zelo hitro razširil. Nastala je velika skupnost z veliko primeri programov, opisov projektov in nasvetov.

Operacijski sistem je naložen na SD spominski kartici, napajanje pa dobi preko mikro USB kabla. Obstajajo trije modeli A, B in B+ (tabela 3.1). Model A je manj, model B pa bolj zmogljiv računalnik. Model B+ pa novejša verzija modela B z nekaj dodatki, ki so jih uporabniki pogrešali pri prejšnjem modelu.

Za Raspberry Pi obstaja veliko programske opreme od operacijskih sistemov do specifičnih projektov. Ravno tako je na voljo veliko dodatkov, kot so baterije, ohišja, kamere, prototipna vezja itd. Zato je zelo primeren za to nalogo.

	Model A	Model B	Model B+
CPE	ARM11 700 MHz		
RAM	258 MB	512 MB	
Ports	HDMI, RCA/composite, Jack 3.5, GPIO		
Network	None	Ethernet 10/100 Mbit/s	
USB port	1	2	4
Onboard storage	SD / MMC / SDIO	MicroSD	

Tabela 3.1: Specifikacije različnih modelov Raspberry Pi



Slika 3.7: Raspberry Pi model B na levi in shema modela B na desni [5].

3.3 Programska oprema

Za delovanje sistema morajo vse naprave imeti svojo programsko opremo. Glavna enota mora imeti uporabniški vmesnik, preko katerega se lahko dostopa do vseh funkcionalnosti sistema. Med vsemi komponentami poteka komunikacija po vnaprej določenih protokolih. Vsako vozlišče posebej mora znati upravljati z napravami, ki so povezane na njo (na primer odklepanje/zaklepanje ključavnice, ali upravljanje s klimatsko napravo). Vsako vozlišče ima namensko programsko opremo, ki je prilagojena in odvisna od mikrokrmilnika, senzorjev, načina komunikacije z glavno postajo in naprave, ki jo to vozlišče upravlja.

3.3.1 OpenHab

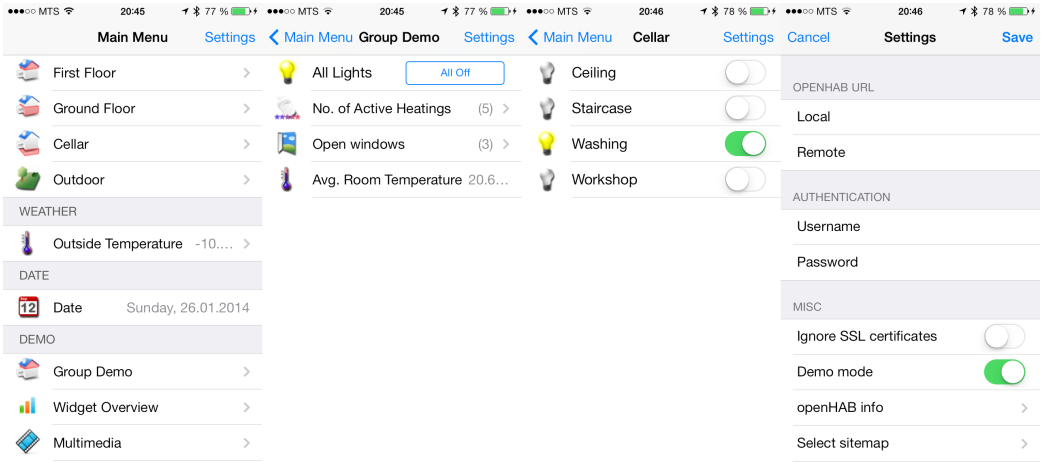
OpenHab je odprto kodna programska oprema, ki ponuja univerzalno integrirano platformo za različne sisteme in tehnologije, ki so uporabljene v pametnih hišah. Temelji na java OSGi ogrodju [13].

OSGi je nabor specifikacij, ki opredeljujejo modularen in dinamičen sistem za Java programski jezik. To omogoči razvijanje manj kompleksnega programa z modularno arhitekturo, dinamično spreminjanje in dodajanje komponent med delovanjem.

OpenHab omogoča povezavo različnih protokolov in tehnologij kot so Bluetooth, Dropbox, GPIO, HTTP, KNX, One Wire, Philips Hue, Serial, TCP/UDP, WOL, Z-Wave. Ponuja enotno uporabniško izkušnjo preko različnih naprav in tehnologij. Uporabnik ima na voljo dostop preko grafičnega vmesnika na pametnih telefonih (Android, iOS) in preko spletnega vmesnika.

Na slikah 3.8, 3.9 in 3.10 so prikazani grafični vmesniki na različnih napravah, preko katerih lahko vidimo stanja posameznih naprav v hiši in/ali jih lahko upravljamo.

Slika 3.8: Grafični vmesnik na Androidu [23].



Slika 3.9: Grafični vmesnik na iOS [23].

Slika 3.10: Grafični vmesnik, ki je dostopen preko spleta [23].

Poglavje 4

Rešitev

Predstavljena je prototipna rešitev, ki vključuje majhno, enostavno, prilagodljivo vozlišče, ki temelji na odprtokodnih zasnovah, idejah in produktih. Vsebuje načine za povezovanje z odprtimi protokoli in tehnologijami. Vse skupaj je povezano s centralno enoto, ki upravlja z vsemi vozlišči. Odprtokodna rešitev s kodo in shemami je na voljo na GitHubu [16].

4.1 OpenHab

OpenHab je nameščen na Raspberry Pi. Pri tem je potrebno paziti, da je strojno računanje podprto v plavajoči vejici znotraj Jave, sicer bo delovalo počasi.

OpenHab je potrebno nastaviti za vsak dom in za vsakega uporabnika posebej. Najprej moramo nastaviti katere naprave imamo v domu in kako z njimi komuniciramo. To naredimo tako, da v konfiguracijsko datoteko <name>.items zapišemo posamezne naprave v formatu:

```
itemtype itemname ["labeltext"] [<iconname>]
        [(group1, group2, ...)] [{bindingconfig}]
```

`itemtype` je tip naprave. Izbiramo lahko med 'barvo', 'datumom', 'številko', 'stikalom', 'besedilom', 'skupino', 'zatemnilnikom', 'kontaktom' in 'senčilom' [14]. `itemName` je ime, ki si ga izberemo. `labelText` je format izpisa, ki bo prikazan. Pomemben je še `bindingconfig`. Tukaj povemo iz kje bo podatek prišel oziroma kam ga bomo pošiljali. Za vsak protokol, tehnologijo oziroma način dostopa

moramo imeti tako imenovani "binding". To je opcijski paket, ki razširja funkcionalnost sistema. Če želimo pridobiti čas s pomočjo "Network Time Protokol", bi nastavitve zapisali v naslednji obliki.

```
DateTime date "Date & Time   [%1$te %1$tb %1$tY
        %1$tk:%1$tM]" [{ntp="Europe/Berlin:de_DE"}]
```

To pomeni, da imamo predmet z imenom `date` tipa `DateTime`. Besedilo, ki se bo prikazalo ima format "`15 Sep 2013 11:42`". Te podatke pa bomo pridobili preko "Network Time Protokol" s parametrom `Europe/Berlin:de_DE`

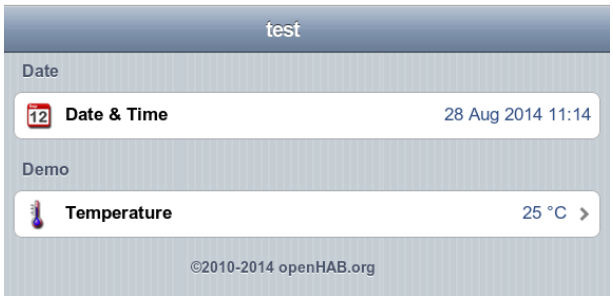
Ko imamo vse predmete definirane, lahko nastavimo kako se bodo prikazali. To nastavimo v datoteki <name>.sitemap.

Primer:

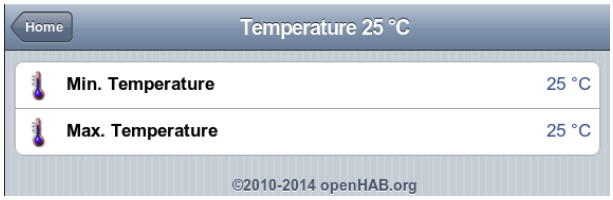
```
sitemap test label="test"
{
   Frame label="Date" {
      Text item=date
   }

   Frame label="Demo" {
      Text item=Temperature {
         Frame {
            Text item=Temp_Max
            Text item=Temp_Min
         }
      }
   }
}
```

Ta primer nam ustvari vmesnik na sliki 4.1, s podstranjo na sliki 4.2, če kliknemo na temperaturo.

Zdaj imamo že delujočo eno od funkcionalnosti v pametni hiši, ki pa še ni preveč pametna. Napisati moramo še nekaj pravil, ki definirajo, kako se pametna hiša odziva na spremembe v okolici oziroma v tem primeru na spremembe v vrednosti



Slika 4.1: Grafični vmesnik zgeneriran iz primera kode.



Slika 4.2: Podstran grafičnega vmesnika zgeneriranega iz primera kode.

predmetov. To povemo tako, da v datoteko <name>.rules zapišemo pogoje v formatu:

```
rule "rule name"
when
    <TRIGGER CONDITION1> or
    <TRIGGER_CONDITION2> or
    <TRIGGER_CONDITION3>
    ...
then
    <EXECUTION_BLOCK>
end
```

Primer, ko prejmemo besedilo ”light on” na predmet VoiceCommand pošljemo ukaz ON na predmet LED:

```
rule "Voice Command"
when
```

```
    Item VoiceCommand received update
then
    var String tmp = VoiceCommand.state.toString.toLowerCase()
    switch(tmp) {
        case "light on":
            sendCommand(LED, ON)
        case "light off":
            sendCommand(LED, OFF)
    }
end
```

Ker Raspberry Pi me more brezžično komunicirati z vozliščem, mu je potrebno dodati Arduino z nRF24L01+ čip. Na Arduino-tu teče enak program kot na vozlišču z razliko, da posluša na serijskem portu. Kar sprejme na serijskem portu pošlje brezžično na pravo vozlišče. Kar pa sprejme brezžično pošlje na serijski port.

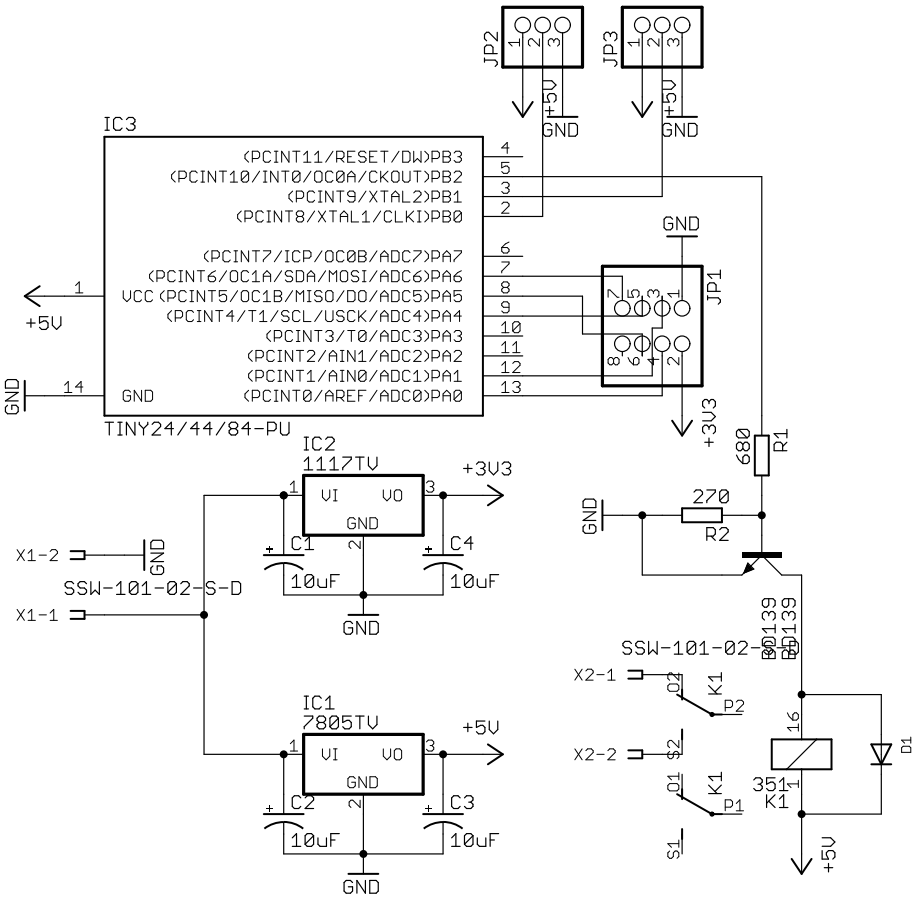
Metoda loop teče neprestano. Ko se pa pojavijo podatki na serijskem vhodu se sproži dogodek serialEvent. Koda:

```
void loop(void)
{
  // Pump the network regularly
  network.update();

  // Is there anything ready for us?
  while ( network.available() )
  {
    // If so, grab it and print it out
    RF24NetworkHeader header;
    char tmp[100];
    network.read(header, tmp, sizeof(tmp));

    char buffer[100];
    sprintf(buffer, "%02X%01X%02X%s\n", header.from_node,
```

```
            header.sensor, header.type, tmp);
    Serial.print(buffer);
  }
}

void serialEvent() {
  delay(10);      // So that all data is available at once

  char tmp[100];
  int i = 0;
  while(Serial.available() && (tmp[i++]=Serial.read()) != '\n');
  tmp[i+1] = 0;

  int to_node, sensor, type;
  sscanf (tmp,"%02X%01X%02X%s", &to_node, &sensor, &type, tmp);


  RF24NetworkHeader header(to_node, type);
  header.sensor = sensor;

  network.write(header, tmp, sizeof(tmp));
}
```

4.2 Vozlišče

Shema vozlišča je na sliki 4.3. Prikazuje vse komponente in priključke, ki sestavljajo vozlišče. Pri razvoju programa je bila uporabljena knjižnica RF24Network [15], ki je napisana za Arduino. Kodo je bilo potrebno spremeniti, optimizirati, da deluje na ATtiny84. Koda vozlišča:



Slika 4.3: Shema vozlišča

```
void loop(void)
{
  // Pump the network regularly
  network.update();

  // Send data every <interval> second
  unsigned long now = millis();
  if ( now - last_sent >= interval  )
  {
    last_sent = now;
    if (DHT11.read(DHT11PIN) == DHTLIB_OK)
```

```
    {
      char tmp[3];
      RF24NetworkHeader header(0, TEMPERATURE_SENSOR_NUM, 127);

      sprintf(tmp, "%d", DHT11.temperature);
      network.write(header, tmp, sizeof(tmp));

      header.sensor = HUMIDITY_SENSOR_NUM;
      sprintf(tmp, "%d", DHT11.humidity);
      network.write(header, tmp, sizeof(tmp));
    }
  }

  // Read available data.  -  Is there anything ready for us?
  while (network.available())
  {
    // If so, grab it and print it out
    RF24NetworkHeader header;
    char tmp[10];
    network.read(header, tmp, sizeof(tmp));

    if (header.sensor == LAMP_SENSOR_NUM) {
      if (tmp[0] == '1') {
         digitalWrite(LAMP_SENSOR_PIN, HIGH);
      } else {
         digitalWrite(LAMP_SENSOR_PIN, LOW);
      }
    }
  }
}
```

V sistem lahko povežemo več vozlišč. Tako lahko imamo več senzorjev, ki so razporejeni po celem domu. Vozlišča so urejena drevesno po naslovih v omrežje. Vsak naslov je 15 bitni. Osmiško zapisano je vrh 00, 01-05 so njegovi otroci, 021

je drugi otrok vozlišča 01, 0321 je tretji otrok vozlišča 021. Tako lahko sporočila pošljemo do vsakega vozlišča, tudi če ni v neposrednem dometu.

Vozlišče periodično pošilja vrednosti iz senzorjev na Arduino v formatu:

```
ID_vozlišča ID_senzorja tip_sporočila sporočilo
```

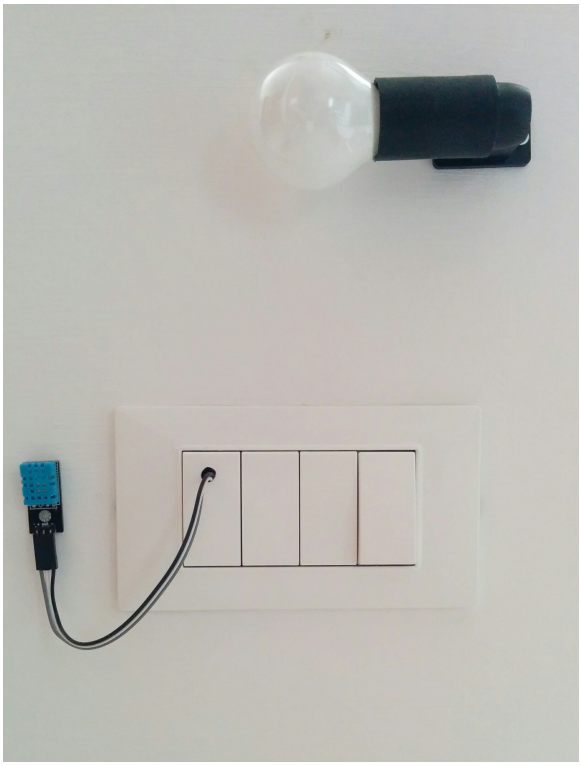
Arduino nato to sporočilo prepošlje preko serijskega protokola na Raspberry Pi v openHab. Ko v openHab-u kliknemo na stikalo se sproži pravilo, ki na serijski protokol pošlje sporočilo z identifikacijo vozlišča in senzorja. Te podatke nato Arduino interpretira in jih pravilno prepošlje do vozlišča kateremu je namenjeno. Vozlišče vklopi/izklopi luč glede na sprejeto sporočilo.

4.3 Prototip

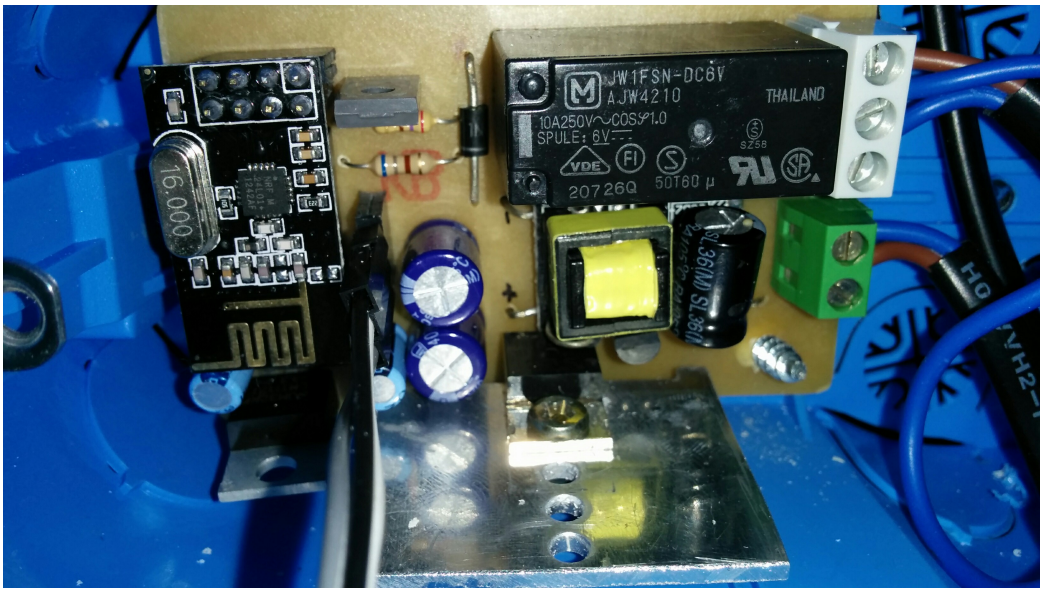
Celotno vezje je vgrajeno v dozo v steno (prikazano na slikah 4.4 in 4.5). Zraven je povezana še navadna žarnica in menjalno stikalo. Stikalo in vezje sta povezana tako, da se lahko s stikalom vklop in z vozliščem izklopi luč ter obratno. Na primer ko je luč vklopljena z vozliščem, se jo s stikalom izklopi in ko je z vozliščem izklopljena se jo s stikalom vklopi (prikaz delovanja na sliki 4.6).

Grafični vmesnik prikazuje trenutni čas, vreme, podatke o temperaturi in vlagi v domu ter stikalo s katerim lahko upravljamo luč (slika 4.7).

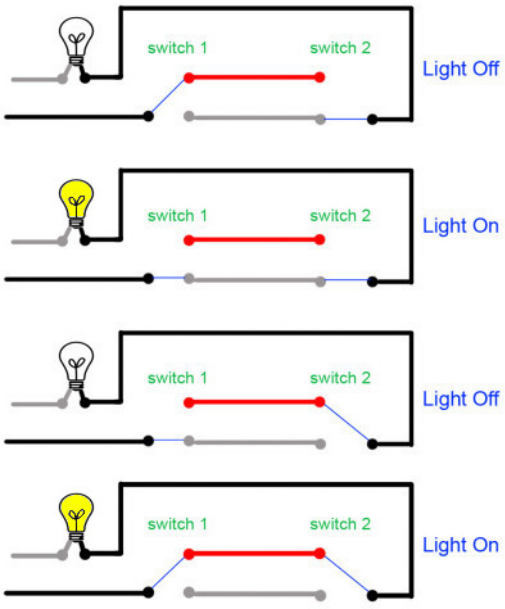
Temperatura in vlaga sta merjeni s senzorjem, ki je izven doze, da se ne meri temperature vozlišča. Ti podatki, se prikažejo v grafičnem vmesniku. Na voljo so podatki o maksimalni in minimalni temperaturi ter grafa zgodovine temperature in vlage. Podatki se beležijo vsako minuto. Slika 4.8 prikazuje podatke o temperaturi in vlagi za zadnje 3 ure.



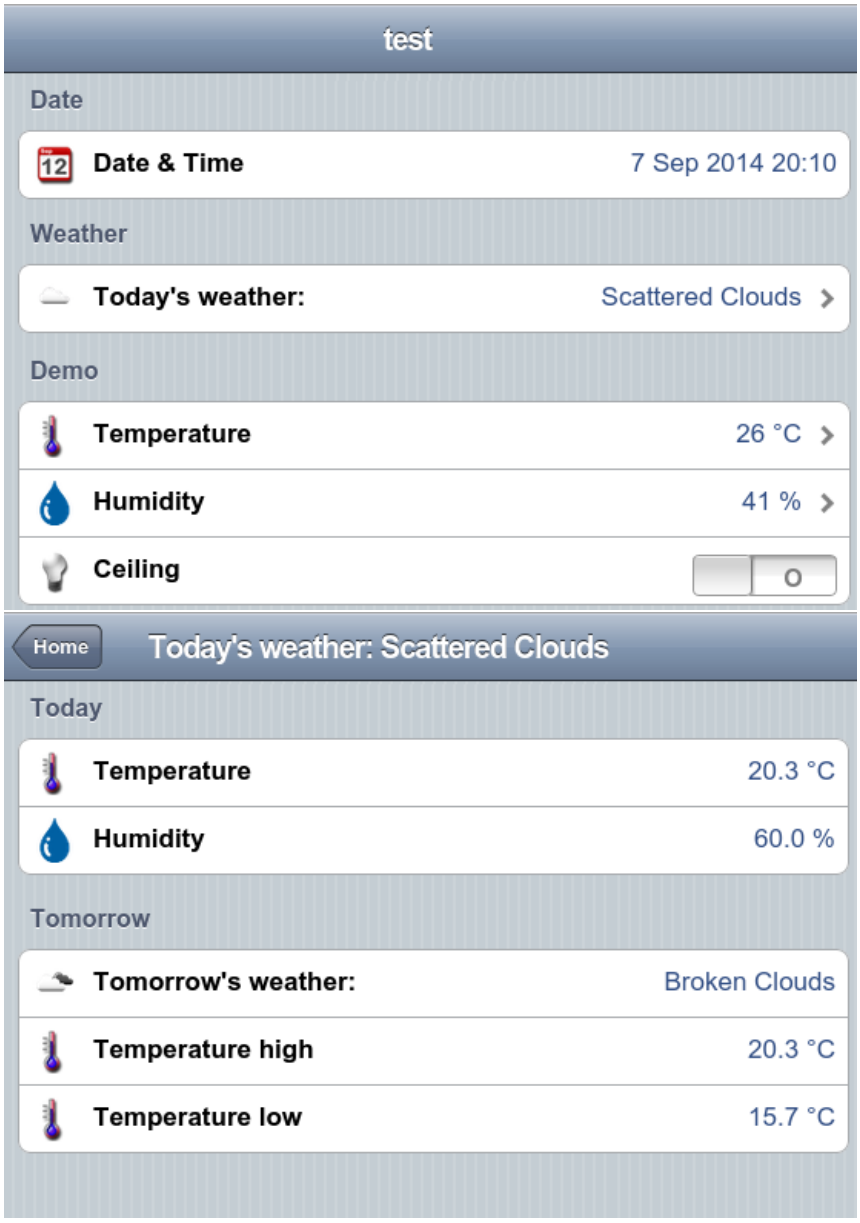
Slika 4.4: Stena s senzorjem in žarnico.



Slika 4.5: Vozlišče vgrajeno v dozo.



Slika 4.6: Prikaz delovanja vezave dveh menjalnih stikal [24].



Slika 4.7: Grafični vmesnik.

Slika 4.8: Prikaz temperature in vlage v grafičnem vmesniku.

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge je nastalo odprtokodno brezžično senzorsko vozlišče. Rešitev zavzema shemo vozlišča, programsko opremo vozlišča in centralne naprave, namestitev v celoten sistem in nastavitev delovanja sistema.

Rešitev je v celoti odprtokodna. Cena takega vozlišča znaša okoli 15€. Kdor koli ga lahko poljubno spreminja, te spremembe objavi na spletu in tako vozlišče izboljšuje in nadgrajuje. Ker je cenovno ugodno in enostavno, ga lahko marsikdo izdela sam doma.

Ena izmed večjih težav, ki še ni rešena, je varnost podatkov pri komunikaciji. Trenutno se vsi podatki pošiljajo kot golo besedilo. Kdorkoli z nRF24 čipom lahko prisluškuje in upravlja z vsemi napravami v domu. Tako se lahko prepriča, da ni nikogar doma in vstopi brez ključa. De neke mere je težava rešljiva s programskim kriptiranjem podatkov. Potrebno pa bi bilo nekatere stvari spremeniti tudi na samem čipu, ki pa so strojno podprte.

Naslednja težava je v vezavi stikala in vozlišča. Stikalo mora biti ločeno od vozlišča, da je naprava (luč) funkcionalna tudi v primeru okvare vozlišča. Če sta vozlišče in stikalo vezana vzporedno bo luč gorela dokler ne izklopimo napajanje s stikalom in z vozliščem. Na primer, če je bila luč prižgana preko vozlišča, je ne bo mogoče ugasniti s stikalom. Če pa je vezano s pomočjo menjalnih stikal, se lahko zgodi, da bo skozi vozlišče tekel tok, ki bo držal luč izklopljeno. Pri tem se lahko pojavi tudi problem pregrevanja.

Sama nastavitev sistema pa ni najbolj enostavna predvsem za tistega, ki se še ni spoprijel s programiranjem. Lahko bi izdelali grafični vmesnik, ki bi avtomatsko

zaznal naprave. Uporabnik bi jih nato poimenoval in z grafičnimi gradniki nastavil pravila.

Literatura

[1] If You Can't Stand the Coding, Stay Out of the Kitchen: Three Chapters in the History of Home — Dr Dobb's. Dostopno na: http://www.drdobbs.com/architecture-and-design/if-you-cant-stand-the-coding-stay-out-of/184404040

[2] Eclipse SmartHome — projects.eclipse.org. Dostopno na: http://projects.eclipse.org/projects/technology.smarthome

[3] Lin, Hsien-Tang, "Implementing Smart Homes with Open Source Solutions", International Journal of Smart Home št. 7, zv. 4, str. 289-295, 2013.

[4] AVR Tutorial - How programming works. Dostopno na: http://www.ladyada.net/learn/avr/programming.html

[5] Raspberry Pi. Dostopno na: http://www.raspberrypi.org/

[6] Overview — USBtinyISP — Adafruit Learning System. Dostopno na: https://learn.adafruit.com/usbtinyisp

[7] ISO/OSI referenčni model. Dostopno na: http://colos.fri.uni-lj.si/eri/RAC_SISTEMI_OMREZJA/html/racunalniska_omrezaja/isoosi_referenni_model.html

[8] debugWIRE - AVR Dragon. Dostopno na: http://www.atmel.com/webdoc/avrdragon/avrdragon.using_ocd_physical_debugwire.html

[9] Readability Activity Indicator. Dostopno na: http://cdn.slashgear.com/wp-content/uploads/2014/03/hue-lux-1.png

[10] HT_goji_smart_lock_sk_140108_16x9_608.jpg (608×342). Dostopno na: http://a.abcnews.com/images/Technology/HT_goji_smart_lock_sk_140108_16x9_608.jpg

[11] HT_goji_smart_lock_sk_140108_16x9_608.jpg (608×342). Dostopno na: http://www.loxone.com/

[12] tinyAVR Microcontrollers. Dostopno na: http://www.atmel.com/products/microcontrollers/avr/tinyavr.aspx

[13] OSGi Alliance — Technology / The OSGi Architecture. Dostopno na: http://www.osgi.org/Technology/WhatIsOSGi

[14] Explanation of Items · openhab/openhab Wiki. Dostopno na: https://github.com/openhab/openhab/wiki/Explanation-of-Items

[15] maniacbug/RF24Network. Dostopno na: https://github.com/maniacbug/RF24Network

[16] GitHub aljosamrak. Dostopno na: https://github.com/aljosamrak

[17] Readability Activity Indicator. Dostopno na: http://www.satsignal.eu/ntp/RaspberryPi+case.jpg

[18] veralite.jpg (522×358). Dostopno na: http://dignan17.com/share/veralite.jpg

[19] jibo-family-robot.jpg (477×700). Dostopno na: http://st1.bgr.in/wp-content/uploads/2014/07/jibo-family-robot.jpg

[20] House-and-sensor.jpg (1000×769). Dostopno na: http://www.lifelinkresponse.com.au/wp-content/uploads/2010/11/House-and-sensor.jpg

[21] 11232-01a.jpg (600×600). Dostopno na: https://cdn.sparkfun.com//assets/parts/6/8/3/7/11232-01a.jpg

[22] IT-NRF24L01-2.jpg (390×283). Dostopno na: http://www.epictinker.com/v/vspfiles/photos/IT-NRF24L01-2.jpg

[23] openHAB - empowering the smart home. Dostopno na: `http://www.openhab.org/`

[24] 2LightSwitches.gif (586×203). Dostopno na: `http://0.tqn.com/w/experts/Electrical-Wiring-Home-1734/2010/03/3-way-switch_4.jpg`